Fast Electronic Digital Image Stabilization

Carlos Morimoto Rama Chellappa
Computer Vision Laboratory, Center for Automation Research
University of Maryland, College Park, MD 20742
carlos@cfar.umd.edu - http://www.cfar.umd.edu/~carlos

Abstract

We present a fast implementation of an electronic digital image stabilization system that is able to handle large image displacements. The system has been implemented in a parallel pipeline image processing hardware (Datacube Max Video 200) connected to a SUN SPARCstation 20/612. Our technique is based on a 2D feature-based multi-resolution motion estimation algorithm, that tracks a small set of features to estimate the motion of the camera. The combination of the estimates from a reference frame is used to warp the current frame in order to achieve stabilization. Experimental results using video sequences taken from a camera mounted on a moving vehicle demonstrate the robustness of the system when processing 15 frames per second.

1. Introduction

Electronic digital image stabilization is the process of removing the unwanted motion from an input video sequence by appropriately warping the images. Image stabilization has been used for the computation of egomotion [11, 6], video compression [7], detection of IMOs [2, 9], and tracking of IMOs [1]. For more natural visualization, vehicle models allow the filtering of the high frequency or oscillatory motion due to irregularities of the terrain from the desired "smooth" motion that would result if the vehicle were running over smooth terrain [4, 12].

Fast implementations of electronic image stabilization algorithms are presented in Hansen *et al.* [5], and Morimoto *et al.* [9]. Hansen *et al.* [5] describe the implementation of an image stabilization system based on a mosaic-based registration technique using a pyramidal hardware (VFE-100). The system uses a multi-resolution, iterative process to estimate the affine motion parameters between levels of Laplacian pyramid images. From coarse-to-fine levels, the optical flow of local patches of the image is computed using a cross-correlation scheme. The motion parameters are then computed by fitting an affine motion model to the flow. These parameters are used to warp the previous image of the next finer pyramid level to the current image, and the refinement process continues until the desired precision is achieved. This scheme, combined with the construction of a mosaic image, allows the system to cope with large image displacements. The VFE implementation is capable of stabilizing images of size 128×120 pixels, with image displacements ranging from ± 32 to ± 64 pixels, at a rate of 10 frames per second [5].

We present a fast and robust implementation of a digital image stabilization algorithm that is based on the 2D model described in [13]. A prototype of this system was presented in [9]. Both systems were implemented in a Datacube Max Video 200, a machine commonly used for real-time image processing. The prototype is able to stabilize 128×120 pixels at a rate of 7 frames per second, with image displacements of up to ± 15 pixels. Improvements on the algorithm allowed us to double the frame rate to approximately 15 frames per second using the same resolution images, with image displacements of up to ± 21 pixels between consecutive frames. This performance, as discussed in Section 3, is similar to that presented in [5].

This paper is organized as follows. Section 2 introduces the image stabilization algorithm and describes how it was implemented. Section 3 shows experimental results of the performance of the system, comparing it with other existing systems, and Section 4 concludes this paper.

[0]The support of ARPA under contract DAAH-0493G049 and of the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) is gratefully acknowledged

2. Electronic Image Stabilization

Our electronic digital image stabilization algorithm is composed of two modules. The *motion estimation* module computes the motion between two consecutive frames by tracking a small set of feature points and then fitting a 2D model to the displacements of the features. After the interframe motion is estimated, the motion parameters are sent to the second module, the *motion compensation* module, which keeps a history of the whole movement of the camera in order to "stabilize" the current frame.

2.1. Motion Estimation

We use a 2D rigid motion model [13] that allows translation, rotation around the optical axis and scaling of the image frames. The transformation between two image frames is defined by:

$$\begin{pmatrix} X_0 \\ Y_0 \end{pmatrix} = \mathcal{S} \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} + \begin{pmatrix} \Delta X \\ \Delta Y \end{pmatrix} \tag{1}$$

where (X_i, Y_i) are the image frame coordinates at time t_i for $i = \{0,1\}$, $(\Delta X \;\; \Delta Y)^t$ is the translation vector measured in the image coordinate system of the frame at t_0 (f_0), Θ is the rotation angle between the two frames and $\mathcal{S}$ is the scaling factor. Notice that $\mathcal{S}$ is inversely proportional to the ratio of the distances between two arbitrary image points at times t_0, t_1. Thus $\mathcal{S}$ can be computed given a set of matched points from both frames, independently of the translation and rotation between them.

To estimate the motion parameters given a set S_i of N feature points in frame f_i and the set S_j of corresponding points in frame f_j, the scaling factor $\mathcal{S}$ is estimated first by computing the ratio of the distances in the feature sets relative to their center of mass [13]. Assuming small rotation, the trigonometric terms in (1) can be linearized to compute the remaining translation and rotation parameters. A system of linear equations is then obtained by substituting all N matched feature pairs into the linearized equations. Each pair introduces two equations, hence the linear system has $2N$ equations and three unknowns (Θ, ΔX, and ΔY), which is solved by a least squares method.

2.2. Automatic Feature Tracking

Before computing the motion parameters, we must acquire and track a small set of feature points. The problem with this approach is that the selected features must belong to regions that suit the 2D motion assumption. In general, distant scene points are appropriate. A heuristic rule to select features on the horizon is currently used. The horizon is a very strong visual cue that is also used by other stabilization techniques [3, 4, 12].

Feature tracking is performed by a multi-resolution scheme. The initial step is to construct a Laplacian pyramid for both frames. A Laplacian pyramid $L[t]$ is formed by combining several reduced resolution Laplacian images of frame f_t. Each level of the pyramid will be denoted by $L[t, l]$, and $L[t, 0]$ will have the Laplacian image of the original frame of size Z (an image of size Z has $Z \times Z$ pixels). The Laplacian image is formed by smoothing the input image with a Gaussian kernel and then convolving the smoothed image with a Laplacian kernel operator. The size of an arbitrary pyramid level l will be given by $\frac{Z}{2^l}$. The levels of the pyramids $L[t-1]$ and $L[t]$ are used from coarse to fine resolutions, where each new processed level contributes to refining the position of the features.

A match within an arbitrary level is obtained by minimizing the sum of square differences (SSD) over a neighborhood (search window) around the candidate matches in frame f_t. The selection of features is performed by thresholding $L[t-1, 0]$. This thresholded image is divided into N vertical partitions, where N corresponds to the number of features to be tracked. Each partition is searched from top to bottom and the top most feature is selected for tracking. The features are scaled-down to the coarsest resolution level $L[t-1, j]$ and used to determine their matches in $L[t, j]$. For a feature $P^j_{t-1}(x, y)$, a search for the minimum SSD is performed in a window of size $S = (2s+1) \times (2s+1)$ centered at the pixel $P^j_t(x, y)$. Let $P^j_t(u, v)$ be the selected matching point of $P^j_{t-1}(x, y)$. Notice that the maximum displacement supported by this search is only s pixels. For the next pyramid level, the coordinates of these pairs are scaled-up by a factor of 2. For the feature $P^{j-1}_{t-1}(2x, 2y)$, the search for the minimum SSD is now performed around the pixel $P^{j-1}_t(2u, 2v)$. This process is repeated until the level 0 is reached. Notice that since the displacement is doubled after every level, the total displacement that this algorithm can handle can be very large even for small values of s.

The use of Laplacian images for feature tracking also helps to reduce the sizes of the SSD windows. The Laplacian operator enhances the regions around the tracked features and smoothes the regions of constant brightness that in general surround the edges. Since the Laplacian operator is rotation invariant, edges of different orientations are equally enhanced so that the Laplacian images are not affected by possible rotations of the input images.

2.3. Motion Compensation

The motion compensation module keeps a history of the interframe motion to remove what is unwanted and compute the warping parameters that will stabilize the current image frame. One of the advantages of electronic image stabilization systems is that motion can be compensated on demand, offering great flexibility by simply modifying some parameters of the compensation module. The way motion is estimated and compensated defines the structure of the stabilization system.

One possible way to stabilize an image stream is to keep a reference frame and compute the motion of the current frame relative to that reference. The problem with this approach is that the overlapping region will eventually become insignificant, hence impossible to compute the motion parameters. We will describe an algorithm that computes the motion parameters between two consecutive frames, which are likely to have good overlapping, thus avoiding the previous problem. Although a reference is still necessary, the system is able to compute the interframe motion even when the reference becomes obsolete.

The block diagram of the stabilization algorithm is shown in Figure 1. The Laplacian $L[t]$ is built from the input image and sent to a delay buffer storage that keeps $L[t-1]$ for the estimation process. The feature detection/tracking module uses $L[t-1, 0]$ to extract features which are used for tracking. After the grid-to-grid matches are found, the motion estimation module computes the interframe motion. The motion compensation block must then compute the total motion of the camera by combining all the interframe estimates over time (since it started running, e.g., $t = 0$). The total motion is used to warp the current frame f_t, and the result is sent to the monitor. Since all frames of the sequence are being warped back to $L[0, 0]$, the resulting sequence is stabilized.

Since features are tracked between consecutive frames, they do not need to be extracted in every frame. They can be extract every other frame instead. Suppose that the system uses the frame f_t to extract features. This set can be used for tracking features backwards between f_t and f_{t-1}. When f_{t+1} is acquired, the same set can be used for forward tracking between f_t and f_{t+1}. Finally, when f_{t+2} comes in, new features must be extracted and the process continues with backward tracking.

This strategy is very simple to implement and just requires a slightly more complex control structure. The forward tracking is done just like described previously, but a few changes must be made for backward tracking.

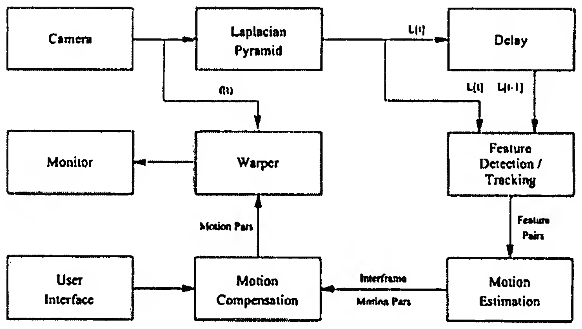

Figure 1. Block diagram of the stabilization algorithm

First, the same tracking algorithm is used by simply switching f_t with f_{t-1} and the motion based on the inverted pairs is computed. The result of this process estimates the motion from f_t to f_{t-1}. A simple transformation is used to obtain the forward motion from the backward motion.

If the motion can be reliably estimated for large feature displacements, it is very simple to extend the backward/forward estimation scheme to utilize larger steps between feature extractions.

Due to the dynamic nature of the problem, the scene as a whole is under constant modification, so that after some relatively short time the original frame might not have anything in common with the current frame. For display purposes, the user should then intervene and reinitialize the system, so that another reference is taken and displayed. This process can be made autonomous by setting limits over the motion parameters, so that whenever the reference frame becomes obsolete, the system can reset itself automatically.

To obtain better visualization for lateral camera motion, mosaic images can be constructed. When the camera moves predominantly parallel to its optical axis, the scaling factor $\mathcal{S}$ may not be compensated for, what produces an output sequence that looks derotated only [3]. The smoothing techniques suggested by [4, 12] could be applied for the remaining parameters.

The system was implemented using a Max Video 200 board connected to a SUN SPARCstation 20/612 via a VME bus adaptor. The MV200 is a parallel pipeline image processing hardware system manufactured by Datacube Inc., very commonly used for real-time image processing.

3. Experimental Results

This section presents experimental results obtained from a video sequence recorded from a camera rigidly mounted on a moving vehicle. Unfortunately, still images are not the most appropriate way of displaying the results of a dynamic process such as stabilization. Those readers with access to www browsers might have a look at **http://www.cfar.umd.edu/~carlos/ stabilization.html**, where a few example sequences are available in MPEG format.

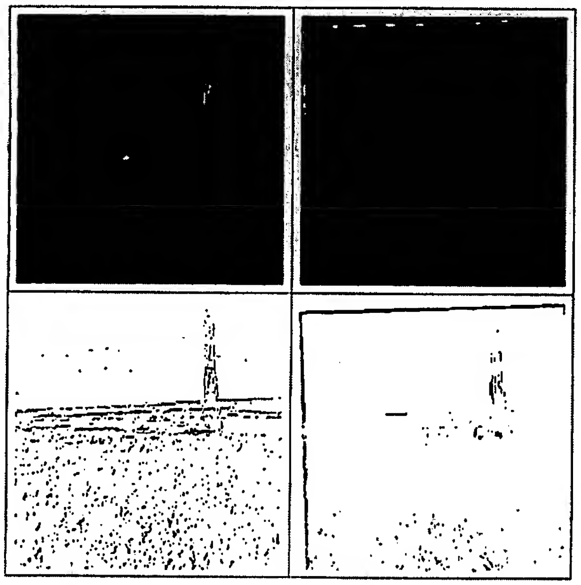

Figure 2. Stabilization results for off-road navigation.

The upper left image of Figure 2 shows frame F of the original sequence. The upper right corner corresponds to the stabilized image of that frame F_t. The bottom left image shows the difference between frame F_t and its previous frame F_{t-1} from the original sequence, and the bottom right shows their difference after stabilization. In both cases, only the pixels with absolute difference higher than 10 are displayed. The dense black region around the difference of stabilized frames are due to the compensation of motion (the image warping performed to register frame F_t to the reference frame). Since the algorithm registers the current frame to the reference, the difference of images can be considered as an error measure that stabilization attempts to minimize.

Observing the image differences, it is clear that the error is quite large in the regions that are closer to the camera, but it is considerably reduced in the regions around the horizon, where the features are extracted and tracked. Actually, the regions closer to the camera do not fit the 2D motion computed from the tracking of features on the horizon, but still, the system is able to process the sequence.

We can compare the performance of our system with the VFE based system presented in [5] by considering the case of pure image translation with constant velocity. In such a case, the VFE system supports velocities of 320 pixels per second, while our Datacube based system supports 313 pixels per second. These figures are important when stabilizing off-road sequences, but sometimes not high enough. The robustness of both systems to support higher displacements can be increased by enlarging the search windows, but with loss in frame-rate speed. The table in Figure 3 shows how the frame-rate of our datacube implementation degrades when the search window is increased and the SSD windows are kept constant (7×7 pixels). Local windows of size 3 pixels allows global displacements of size 21 (at the finest resolution). At this settings the system can process 14.9 frames per second. The last row of the table shows the performance of the VFE implementation and the right-most column gives the maximum pixel velocities that each setting is able to process. This might suggest that our system is more robust when set to search displacements of magnitude 5 instead of 3 because despite the loss in frame-rate there is a considerable gain in feature velocity that it is able to track. Unfortunately, when the search space is increased, the narrow SSD window being used is more likely to find a false match. Increasing the SSD window provides better discrimination between features and decreases the probability of false matches. On the other hand the frame-rate drops considerably.

Local	Global	Rate	Velocity
±3	± 21	14.9	312.9
±4	± 28	12.6	352.8
±5	± 35	10.5	367.5
VFE	± 32	10	320

Figure 3. Performance evaluation of search window sizes.

One of the advantages of our approach is that the matches are restricted to distant points and not blocks distributed all over the image like the VFE system. This allows our system to work under situations where regions, which are not used to track features, present very intense flow that do not fit the 2D rigid motion assumption. When the flow is due to dominant translation, the displacement of distant features will be smaller than the displacement of close features. Thus using a subset of feature pairs with the smallest displacements to estimate the motion parameters contributes to enhance the robustness of the system. A possible alternative to the current feature extraction and selection algorithm would be to use a fast and robust optical flow algorithm, such as the one presented in [8], to select the regions where the optical flow is small and then constrain the search for features to these regions.

Although the resulting stabilized sequence is appropriate for visualization purposes, it cannot be directly applied to segment IMOs from simple difference of frames, mainly due to the clutter produced by the regions that do not fit the 2D motion assumption and small misalignments of the registration process, but observe that the moving object present in the sequence is clearly segmented. Qualitative methods [10] and temporal filters combined with velocity tuned filters [9] were suggested to detect IMOs in such conditions.

4. Conclusion

We presented in this paper a fast electronic digital image stabilization system based on a two-dimensional feature-based multi-resolution motion estimation algorithm, that tracks a small set of features to estimate the motion of the camera. Stabilization is achieved by combining all motion from a reference frame and subtracting this motion from the current frame. The system was implemented in a Datacube Max Video 200 board connected to a SUN SPARCstation 20/612 via a VME bus adaptor. Preliminary tests using video sequences recorded from a vehicle moving on rough terrain demonstrate the robustness of the system, that is able to process 15 frames per second and handle displacements of up to ± 21 pixels between consecutive frames. Although these figures might be compatible or better than current stabilization systems, it is sometimes not enough to stabilize sequences taken in off-road conditions.

Further research effort is under way to integrate stabilization with other image processing applications like video compression and automatic detection, tracking, and recognition of IMOs.

Acknowledgements

We would like to thank the Laboratório de Sistemas Integráveis of the University of São Paulo (LSI - USP) for their kind support in the preparation of this paper.

References

[1] S. Balakirsky. Comparison of electronic image stabilization systems. Master's thesis, Department of Electrical Engineering, University of Maryland , Collage Park, 1995.

[2] P. Burt and P. Anandan. Image stabilization by registration to a reference mosaic. In *Proc. DARPA Image Understanding Workshop*, pages 425–434, Monterey, CA, November 1994.

[3] L. Davis, R. Bajcsy, R. Nelson, and M. Herman. Rsta on the move. In *Proc. DARPA Image Understanding Workshop*, pages 435–456, Monterey, CA, November 1994.

[4] Z. Durić and A. Rosenfeld. Stabilization of image sequences. Technical Report CAR-TR-778, Center for Automation Research, University of Maryland, College Park, 1995.

[5] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In *Proc. DARPA Image Understanding Workshop*, pages 457–465, Monterey, CA, November 1994.

[6] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 454–460, Seattle, WA, June 1994.

[7] O. Kwon, R. Chellappa, and C. Morimoto. Motion compensated subband coding of video acquired from a moving platform. In *Proc. of IEEE International Conf. on Acoustics, Speech, and Signal Processing*, pages 2185–2188, Detroit, MI, January 1995.

[8] H. Liu, T. Hong, M. Herman, and R. Chellappa. A general motion model and spatio-temporal filters for computing optical flow. Technical Report CAR-TR-741, Center for Automation Research, University of Maryland, College Park, October 1994.

[9] C. Morimoto, D. DeMenthon, L. Davis, R. Chellappa, and R. Nelson. Detection of independently moving objects in passive video. In I. Masaki, editor, *Proc. of Intelligent Vehicles Workshop*, pages 270–275, Detroit, MI, September 1995.

[10] R. Nelson. Qualitative detection of motion by a moving observer. *International Journal of Computer Vision*, 7:33–46, 1991.

[11] T. Viéville, E. Clergue, and P. Facao. Computation of ego-motion and structure from visual and internal sensors usig the vertical cue. In *Proc. International Conference on Computer Vision*, pages 591–598, Berlin, Germany, 1993.

[12] Y. Yao, P. Burlina, and R. Chellappa. Electronic image stabilization using multiple visual cues. In *Proc. International Conference on Image Processing*, pages 191–194, Washington, D.C., October 1995.

[13] Q. Zheng and R. Chellappa. A computational vision approach to image registration. *IEEE Trans. Image Processing*, 2:311–326, 1993.